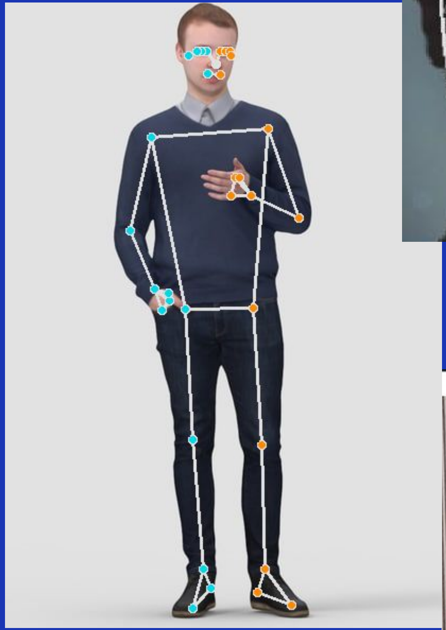
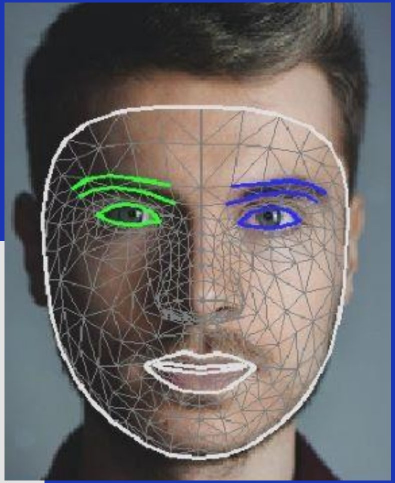
Toolbox 2

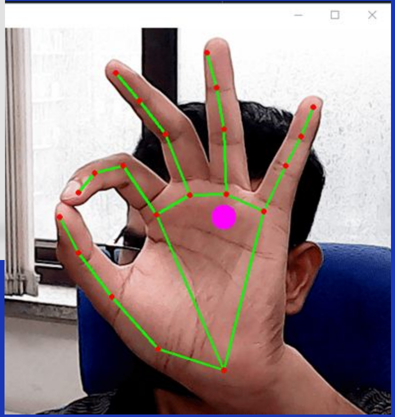
MediaPipe

March 27, 2025

Antoine Assaf







What is MediaPipe?

- Developed by Google, MediaPipe is a real-time human model estimation tool
- Uses TensorFlow Lite to run CNN on-device
- Can be imported into Python, but its core is written in C++ for efficiency
- Will focus on MediaPipe Pose specifically, human poses by mapping a 33-segmented human model into 3D coordinates from a 2D image

Mediapipe Pose "Pipeline"

Output capture from Open CV

OpenCV Output: B G R

Open Computer Vision Library

-OpenCV (Open Computer Vision Library) is fed with an image I provide which stores it in B G R

- MediaPipe (Pose, Object, Hand, Face Tracking) all require R G B format for color channels

```
import cv2

rgb_image = cv2.cvtColor(bgr_image, cv2.COLOR_BGR2RGB)
```

Now the image is in R G B , which is passed into process function which returns a landmark object

```
results = pose.detect(rgb_image)
```

Great but

(1) how does MediaPipe's detect function work?

(2) And how to parse the output, making it's useful for computational physics?

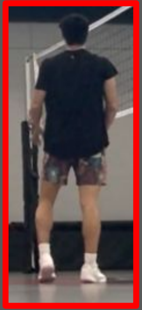
Mediapipe Pose Detect function

Multi-step NN process

First NN "BlazePose Detector" Computes Region of Interests

ROI 1

ROI 2



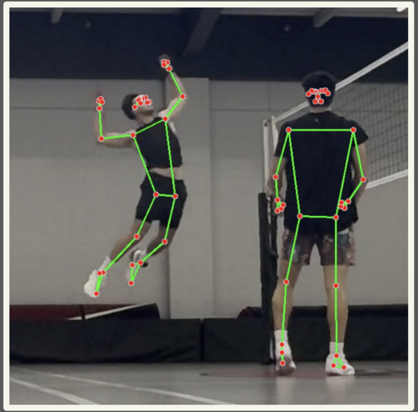
$$I \in \mathbb{R}^{256\times256\times3}$$

More preprocessing

- **Downscaling to 256×256×3**
- **Pixel Normalization**

$$f^{(l)} = \sigma(W^{(l)} * f^{(l-1)} + b^{(l)})$$

Second Larger TFLite CNN Model
Predicts Pose



CNN Direct Regression Output per ROI (33 landmarks in each rig, more this next)

This TFLite Models run on CPU/GPU:

Identifies multiple ROI's (if existing)

- Head + torso region
- Used to crop + normalize the person's body
- Only called when ROI is lost (useful for video)

CNN Direct Regression Output per ROI (33 landmarks in each rig, more this next)

$$\hat{Y} = \begin{bmatrix} x_1 & y_1 & z_1 & v_1 \\ x_2 & y_2 & z_2 & v_2 \\ \vdots & \vdots & \vdots & \vdots \\ x_{33} & y_{33} & z_{33} & v_{33} \end{bmatrix} \in \mathbb{R}^{33\times4}$$

Detect function output:

$$\hat{\mathbf{p}}_i = (\hat{x}_i, \hat{y}_i, \hat{z}_i, \hat{v}_i)$$

$$\hat{Y} = \begin{bmatrix} x_1 & y_1 & z_1 & v_1 \\ x_2 & y_2 & z_2 & v_2 \\ \vdots & \vdots & \vdots & \vdots \\ x_{33} & y_{33} & z_{33} & v_{33} \end{bmatrix} \in \mathbb{R}^{33\times 4}$$

- 33 “**landmarks**” (red dots)
- x, y positions [0, 1]
- z position ... (-inf to inf)

→ used for relative landmark ordering

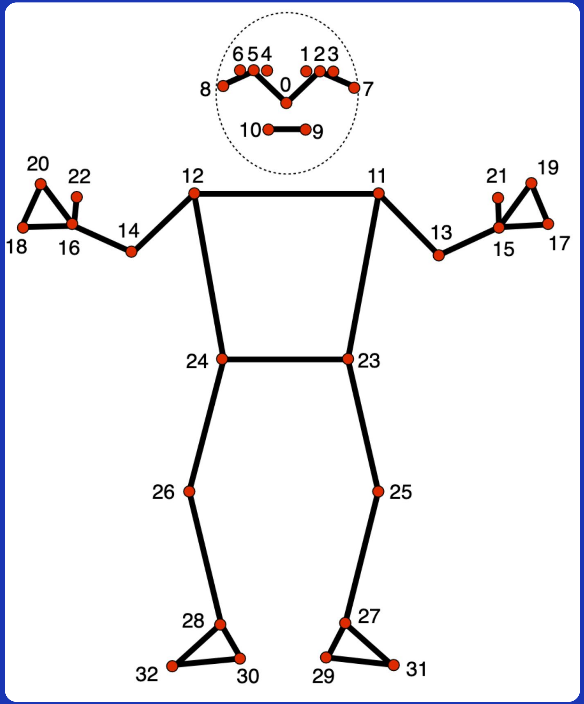
→ pelvis z ≈ 0

→ closer to camera, z < 0

→ farther from camera, z > 0

- v visibility score [0, 1]

→ confidence level for how visible the joint is seen on screen



OK Great, but modeling just one image is not that useful for physics problems

Let's see how OpenCV and Mediapipe can incorporate the temporal element …

Videos with Mediapipe

MediaPipe adds a temporal component

- It reuses the previous frame's landmarks to estimate the next frame's ROI
- The Pose Detector is skipped most frames to improve speed
- smooth, continuous pose tracking with additional temporal filters automatically added by MP in Video mode

Ex: Landmark Smoothing 1€ Filter

$$\hat{x}_t = \alpha \cdot x_t + (1 - \alpha) \cdot \hat{x}_{t-1}$$

- **α** adaptive smoothing factor (depends on velocity of landmark)
- **x_t** predicted landmark position
- **X_{t-1}** previous landmark position

MediaPipe Flexibility

MediaPipe Pose

Can adjust parameters to match needs

```
base_options = python.BaseOptions(model_asset_path='pose_landmarker.task')
options = vision.PoseLandmarkerOptions(
    base_options=base_options,
    num_poses=2,
    output_segmentation_masks=True,
    min_pose_detection_confidence = .5,
    min_pose_presence_confidence = .5,
    min_tracking_confidence = .5)
detector = vision.PoseLandmarker.create_from_options(options)
```

** You can also train the CNN to detect any model with any rig by supplying photos expected outputs through MediaPipe

Other MediaPipe Trained Models

Hands (21 landmarks per hand)

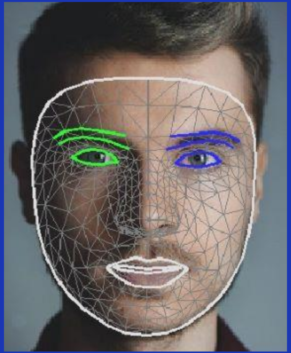
Face Mesh (468 face landmarks, 3D head pose)

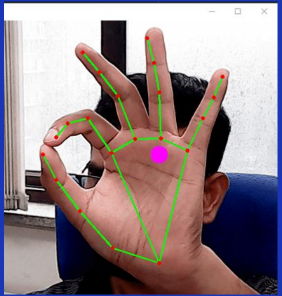
Objectron (3D object detection and tracking for common objects)

Object Detection (2D bounding boxes for general objects)

Box Tracking (tracking custom boxes across frames)

Instant Motion Tracking (for AR-like effects)





MediaPipe Physics Use Cases

Many use cases, but to name a few:

Mediapipe Pose

- Human Motion Analysis in sports, to get joint positions, angles, velocities to compare with simulations, to optimize performance
- Biomechanics Modeling (Full-body motion for inverse kinematics analysis)
- Energy Transfer Analysis (Comparing limbs before/after impact

Object Trajectory Tracking

- Tracking position of a thrown object across frames
- Great solution if object doesn't have sensors or trajectory would be impacted with one

Rigid Body Motion 3D Object Orientation

- Analyze 3D position & orientation of an object

You can also use MP to overlay tracked motion vs predicted trajectory